Advanced Python in HEP Starterkit 2021

Jonas Eschle

… building on the existing starterkit material

Installing Python

- **Recommended: use anaconda/miniconda, always with environments**
  - Package installer that can handle way more:
    - Install ROOT (within few minutes, no kidding!)
    - Have multiple Python and package versions

Attention:
- can grow big (~GB)! Install to "data" folder with still fast I/O rate, not home
- Disable base environment (see installer)
- Add conda-forge to channels

Let's dive in!

Python

- **Interpreted language**
  - Every line gets compiled, then executed
    → no need for pre-compilation
  - Very dynamic, a lot of user-friendly features
- **#1 language in Data Analysis, most popular language in general**
- **General purpose programming language**
- **Heavily used (guesstimated > 50%) inside LHCb, CMS, others...**

Python in short

- **Python is easy**

- **Python is slow**

Python in short

- **Python is easy**
  - Simple to learn, just use it

- **Python is slow**
  - Unfortunately, therefore C++ must be used for large data

Python in short

- ~~**Python is easy**~~
  - ~~Simple to learn, just use it~~

"Driving a car is easy. Just push the power. Any 5 year old can do..."

Python in short

- ~~**Python is easy**~~
  - ~~Simple to learn, just use it~~

"Driving a car is easy. Just push the power. Any 5 year old can do..."

(Which is "true" compared to e.g. a bicycle)

Python in short

- ~~**Python is easy**~~
  - ~~Simple to learn, just use it~~

"Driving a car is easy. Just push the power. Any 5 year old can do..."

Problem: accidents <=> bugs
(see e.g. this talk)

Python in short

- **Python is beautiful!**
  - Clean, powerful and well designed
  - ... and yes, less to care about

Python in short

- **Python is slow**
  - Unfortunately, therefore C++ must be used for large data

Python in short

- ~~**Python is slow**~~
  - ~~Unfortunately, therefore C++ must be used for large data~~

Python in short

A Ferrari is slow...

- ~~**Python is slow**~~
  - ~~Unfortunately, therefore C++ must be used for large data~~

Python in short

A Ferrari is slow...

True, if you transport goods, compared to a truck

- ~~**Python is slow**~~
  - ~~Unfortunately, therefore C++ must be used for large data~~

Python in short

"Writing C++ code, that does heavy number crunching, with Python syntax and executing it with a Python interpreter is slow"

True!

- ~~**Python is slow**~~
  - ~~Unfortunately, therefore C++ must be used for large data~~

Python in short

Python is a high
level language

Python

C++

Compiler and more

- ~~**Python is slow**~~
  - ~~Unfortunately, therefore C++ must be used for large data~~

Python in short

- **Python is a fast, high level language**
  - It uses C++/Fortran code for computations
  - No need for manual implementation
    e.g. just add vectors

Python in short

- **Python is beautiful!**
  - Clean, powerful and well designed
  - … and yes, less to care about

- **Python is a fast, high level language**
  - It uses C++/Fortran code for computations
  - No need for manual implementation
    e.g. for vector operations

Python in HEP

- **User base:**
  - Many C++ users, Python still "newish" (and Python-writing-in-C++) → not every script is perfect Python
- **Very useful for data analysis**
  - Shared ecosystem with open-source community
  - HEP ecosystem new, strong in last few years
- **Many packages in Python for HEP → see Scikit-HEP**
- **ROOT can be used with Python**

This lecture

- **Ideal: fully planned lectures, well designed, perfect pace...**

This lecture

- ~~Ideal: fully planned lectures, well designed, perfect pace…~~
- Everyone has a different Python level (and it's spread usually)
- *average* level of Python increases every year
- Python ecosystem changes *rapidly*

This lecture

- ~~Ideal: fully planned lectures, well designed, perfect pace…~~
- Everyone has a different Python level (and it's spread usually)
- *average* level of Python increases every year
- Python ecosystem changes *rapidly*

Focus is on HEP in Python, there are some tutorials, but…

let's find out together!

This lecture is...

- **...dynamic. Depending on you.**
  - More material available than time
  - Follows *somewhat* online course
- **...not just an overview**
  - You will spend a lot of time with Python
  - Some deeper understanding of critical parts
- **...interactive.**
  - Solve small problems and ask about the libraries
  - Ask questions about concepts and code

This lecture is...

- **(Introduction to classes?)**
- **Medium Python concepts**
  - *, **; context manager, exceptions,...
  - (numpy, math libraries?)
- **Basics of data handling and plotting**
  - Uproot, pandas, cuts, plots...
- **Multivariate analysis (machine learning)**
  - BDT, uniformity, training and test
- **Histograms and reweighting**
- **Likelihood fits and inference**
  - Building models, fitting, significance
- **Scikit-HEP diverse tools**

Editors

- **Vim or Emacs**
- **Simple editors like nano, gedit**
- **IDE (Integrated Development Environment)**
  - Offers great support for so many things
  - Special mention: PyCharm

Learning a language

Languages

- **Programming similar to natural language**
- **How to learn:**
  - Grammar/Syntax + Rules
  - Practice
  - Time
- **Be curious!**
  - Ask yourself: what *exactly* is happening?
  - Try out: what if I do this?

Conventions

- **Can I write...**
  - Hel lo, i havearri ved
  - Хелло, и хаве арривед.
  - Hallo, ich bin angekommen.

  ... if I wanna say "Hello, I have arrived."?

Conventions

- **Can I write...**
  - Hel lo, i havearri ved
  - Хелло, и хаве арривед.
  - Hallo, ich bin angekommen.

  ... if I wanna say "Hello, I have arrived."

No!

Conventions

- **Can I write...**
  - Hel lo, i havearri ved
  - Хелло, и хаве арривед.
  - Hallo, ich bin angekommen.

  ... if I wanna say "Hello, I have arrived."
- **Conventions:**
  - When there are several ways, we agree on one
  - Make life easier for everyone!

There should be one – and preferably only one – obvious way to do it.

Python conventions

- **We are *very* picky…**
  - Hello, i have arrived.
  - LHCb papers ask for BE instead of AE

Python conventions

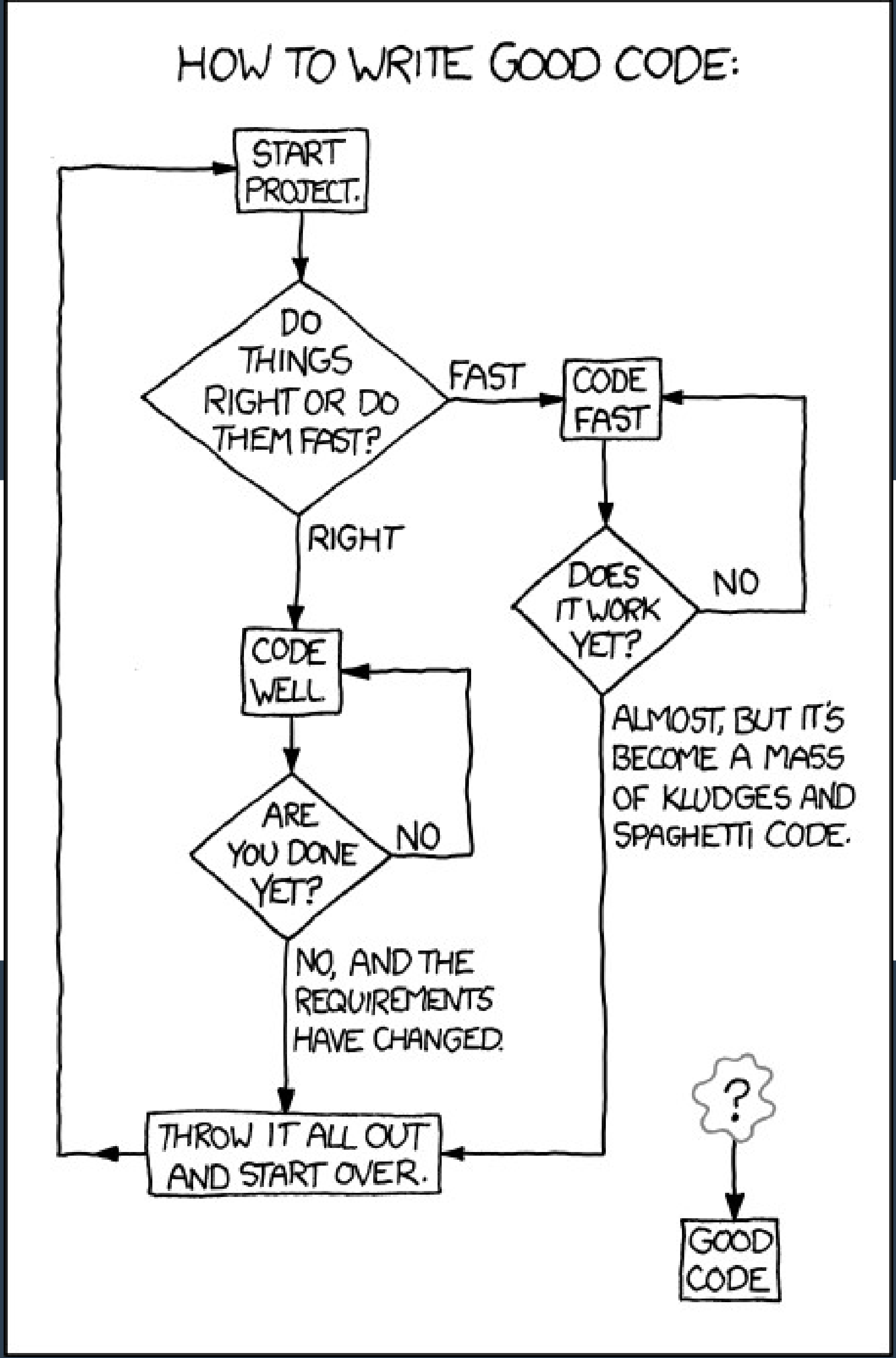
- **We are *very* picky...**
  - Hello, i have arrived.
  - LHCb papers ask for BE instead of AE
- **Python lives from conventions**
  - Because it is so powerful and let's you do a lot
    "We are consenting adults"
  - PEP8: style-guide (many auto checkers)
  - Google for the "best" (most pythonic) solution

Python conventions

- **We are *very* picky...**
  - Hello, i have arrived.
  - LHCb papers ask for BE instead of AE
- **Python lives from conventions**
  - Because it is so powerful and let's you do a lot
    "We are consenting adults"
  - PEP8: style-guide (many auto checkers)
  - Google for the "best" (most pythonic) solution

```
>>> import this
```

writing code is more than writing code


HOW TO WRITE GOOD CODE:
START PROJECT.
DO THINGS RIGHT OR DO THEM FAST?
FAST
CODE FAST
RIGHT
CODE WELL
DOES IT WORK YET?
NO
ALMOST, BUT IT'S BECOME A MASS OF KLUDGES AND SPAGHETTI CODE.
ARE YOU DONE YET?
NO
NO, AND THE REQUIREMENTS HAVE CHANGED.
THROW IT ALL OUT AND START OVER.
?
GOOD CODE

I am not a book author...

...(although I speak English!)

- **A book needs a story...**
  - ... and a language to "implement it"

I am not a book author...

...(although I speak English!)

- **A book needs a story...**
  - ... and a language to "implement it"
- **Book author ↔ software engineer**
  - Knowing how to program ≠ knowing the language

I am not a book author...

...(although I speak English!)

- **A book needs a story...**
  - ... and a language to "implement it"
- **Book author ↔ software engineer**
  - Knowing how to program ≠ knowing the language
- **OOP, composition, interfaces, protocols, CI/CD, unittests, responsibilities, stateful, VCS, code review, legacy, open-close, forward/backwards compatibility, DRY, etc.**

I am not a book author...

...(although I speak English!)

- **OOP, coupling, interfaces, protocols, CI/CD, unittests, responsibilities, stateful, VCS, code review, legacy, forward/backwards compatibility, DRY, etc.**

We do not need to know all that, but:

- Be aware of your limits
- We do not need (and should not!) reinvent programming
  ... it's invented, let's learn from the right people.

Vectorization

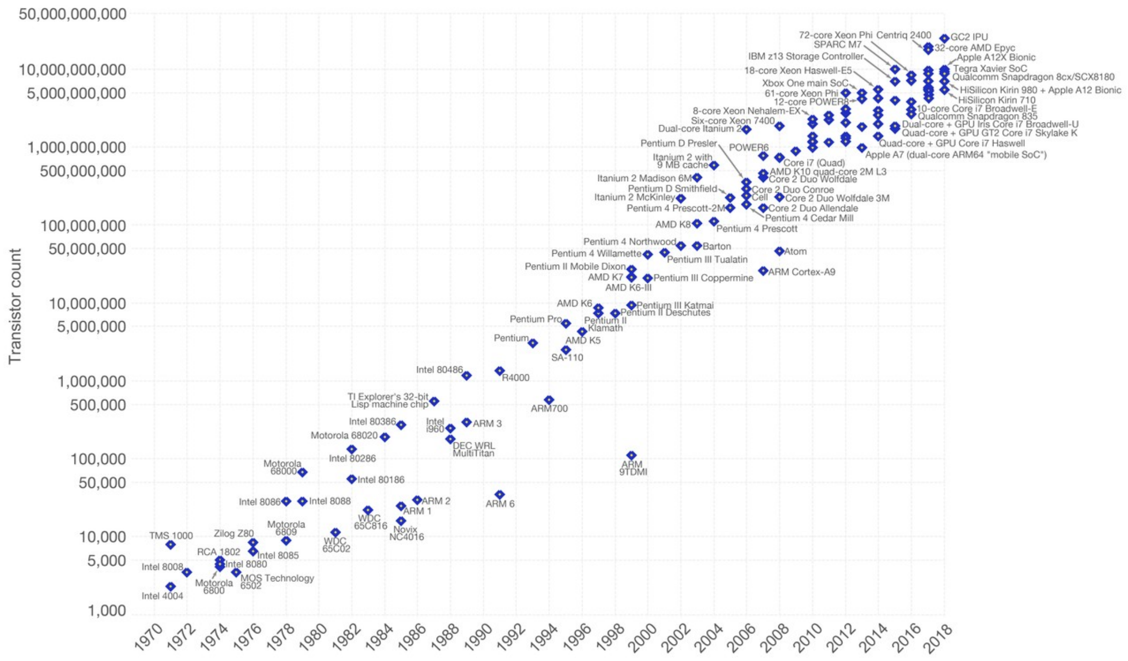
Moore's law

- **every 1.5 year, the CPU power doubles**
- **Stated in the 70s → just holds**

Moore's Law – The number of transistors on integrated circuit chips (1971-2018)

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years. This advancement is important as other aspects of technological progress – such as processing speed or the price of electronic products – are linked to Moore's law.

Our World in Data



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
The data visualization is available at OurWorldinData.org. There you find more visualizations and research on this topic.

Licensed under CC-BY-SA by the author Max Roser.

Well, actually...

- **It states that the clock frequency doubles**
- **One problem: heat dissipation ~ f^3**
  - Not a problem until 2000's
  - Now a problem: frequency stuck at GHz
- **But: parallelization helps (Moore's law still basically true)**

Paralellization

- **You need many exercises to split amongst students**
- **100 students but giving them 1 exercise (serial) after the other is useless**
- **Idea: think vectorized**
  - Vector operations automatically include multiple operations at once
  - Hardware like GPU built for massively parallel execution of the same operation on points

Sometimes we can cheat

- **Some libraries can (sometimes) convert for-loops to vectors**
- **Learn to think vectorized!**
  - (not so easy if used to for-loops)
- **Code vectorized**
  - Pandas, numpy, TensorFlow etcetcetc.
- **Key to speed**

Number crunching!